**ALGORITHMS & THEIR FLOW CHARTS**

PASSWORD CHECKER FUNCTION v1

Description: An algorithm that analyze the password and determine if it is weak according to two criteria which are the length of the password and variety of the characters in the password.

**Plain Text:**

1. Start
2. Parameters: password (String)
3. Declare

* isWeak (Boolean)
* symCounter (integer)
* letterCounter (integer)
* numCounter (integer)

1. Parse the password string into characters. (clue: you can use the unpack ‘#<stringname>’ method here if you’ll write the algorithm with python)
2. Create a new array (list in python) named ‘passwordList’ to assign these characters to it in order to analyze letter by letter

variety of the characters in the password

1. Declare an iterator (i) to traverse in the passwordList for the loop in the next step
2. Condition (If): If the iterator is smaller than length of the passwordList array;

* If true, then go to the 8th step
* Else, then go to the 10th step

1. Condition (If): If decimal number of the char (iterator. element of the passwordList) is in the range of decimal equivalents of symbols in the ASCII table;

* If true, then go to the 9th step after increased one the symCounter
* Else, then skip to the Condition of Else If

Condition (Else If): If decimal number of the char (iterator. element of the passwordList) is in the range of decimal equivalents of letters in the ASCII table;

* If true, then go to the 9th step after increased one the letterCounter
* Else, then skip to the Condition of Else

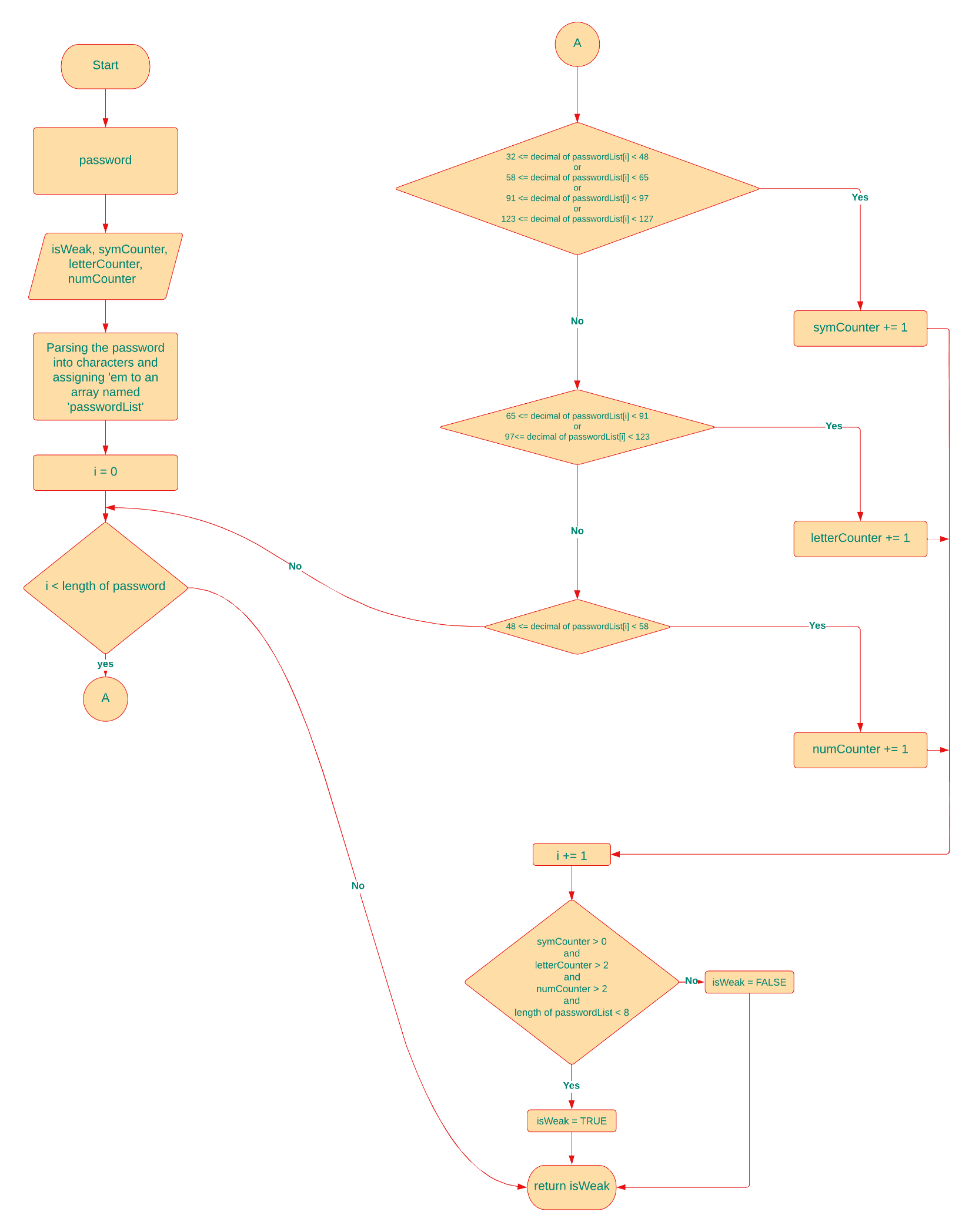
Condition (Else): If decimal number of the char (iterator. element of the passwordList) is in the range of decimal equivalents of numbers in the ASCII table;

* If true, then go to the 9th step after increased one the numCounter
* Else, then 7th step

1. Increase one the iterator
2. When the loop has finished, check if the symCounter bigger than ‘0’ and the letterCounter bigger than ‘2’ and the numCounter bigger than ‘2’ and the length of the passwordList is bigger than ‘8’;

* If true, then isWeak = FALSE (boolean)
* Else, then isWeak = TRUE (boolean)

1. Return isWeak

**Flow Chart:**

PASSWORD CHECKER FUNCTION v2

Description: An algorithm that analyze the password and determine if it is weak according to three criteria which are the length of the password,variety of the characters in the password and the situation of proximity between these different type characters.

**Plain Text:**

1. Start
2. Parameters: password (String)
3. Declare

* isWeak (Boolean)
* symCounter (integer)
* letterCounter (integer)
* numCounter (integer)

1. Parse the password string into characters. (Clue: you can use the unpack ‘#<stringname>’ method here if you’ll write the algorithm with python)
2. Create a new array (list in python) named ‘passwordList’ to assign these characters to it in order to analyze letter by letter

variety of the characters in the password

1. Declare an iterator (i) to traverse in the passwordList for the loop in the next step
2. Condition (If): If the iterator is smaller than length of the passwordList array;

* If true, then go to the 8th step
* Else, then go to the 10th step

1. Condition (If): If decimal number of the char (iterator. element of the passwordList) is in the range of decimal equivalents of symbols in the ASCII table;

* If true, then go to the 9th step after increased one the symCounter
* Else, then skip to the Condition of Else If

Condition (Else If): If decimal number of the char (iterator. element of the passwordList) is in the range of decimal equivalents of letters in the ASCII table;

* If true, then go to the 9th step after increased one the letterCounter
* Else, then skip to the Condition of Else

Condition (Else): If decimal number of the char (iterator. element of the passwordList) is in the range of decimal equivalents of numbers in the ASCII table;

* If true, then go to the 9th step after increased one the numCounter
* Else, then 7th step

1. Increase one the iterator

the situation of proximity between these different type characters

1. Declare an iterator (i) to traverse in the passwordList for the loop in the next step, three Boolean variable named as c1, c2 which are condition 1, 2 and proximity
2. Condition (If): If the iterator is smaller than length of the passwordList array;

If true, then go to the 12th step

Else, then go to the 14th step

1. Condition (If): If decimal number of the char (iterator).element of passwordList is the **same** as decimal number of the char (iterator+1).element;

* c1 = true

Condition (Else If): If decimal number of the char (iterator).element of passwordList is the **same** as decimal number of the char (iterator+2).element;

* c2 = true

Condition (If): If both results of the c1 and the c2 are **true**;

* proximity = true

Condition (Else):

* Condition (If): If c1 = true && c2 = false
  + proximity = true
* Condition (Else If): If c1 = false && c2 = true
* proximity = false
* Condition (Else):
* proximity = false

1. Increase one the iterator
2. When both loops have finished; check if the symCounter bigger than ‘0’ and the letterCounter bigger than ‘2’ and the numCounter bigger than ‘2’ and the length of the passwordList is bigger than ‘8’, proximity = false;

* If true, then isWeak = FALSE (boolean)
* Else, then isWeak = TRUE (boolean)

1. Return isWeak